University of Maryland College Park
Dept of Computer Science
CMSC131 Spring 2011
Midterm II

First Name (PRINT): ______________________________________________________

Last Name (PRINT): _____________________________________________________

University ID: ___________________________________________________________

Section/TAName: _________________________________________________________

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: __________________________________________________________________________

Instructions

➢ This exam is a closed-book and closed-notes exam.
➢ Total point value is 100 points.
➢ The exam is a 50 minutes exam.
➢ Please use a pencil to complete the exam.
➢ WRITE NEATLY. If we cannot understand your answer, we will not grade it (i.e., 0 credit).

Grader Use Only

<table>
<thead>
<tr>
<th>#</th>
<th>Problem</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Problem 1 (General Questions)</td>
<td>(16)</td>
</tr>
<tr>
<td>#2</td>
<td>Problem 2 (Memory Map)</td>
<td>(8)</td>
</tr>
<tr>
<td>#3</td>
<td>Problem 3 (Parsing/Exception)</td>
<td>(20)</td>
</tr>
<tr>
<td>#4</td>
<td>Problem 4 (Class Definition)</td>
<td>(56)</td>
</tr>
<tr>
<td>Total</td>
<td>Total (100)</td>
<td>(100)</td>
</tr>
</tbody>
</table>
Problem 1  (16 pts)

1. (1 pt) Name one class discussed in class that is immutable. ________________

2. (1 pt) When should we define a method as static?

3. (1 pt) What is the default value of reference instance variables of a class? ________________

4. (2 pts) When is space for a local integer variable allocated and when is it recovered?

5. (2 pts) Why do we never use == to compare floating point numbers?

6. (1 pt) When is the finally block associated with exceptions executed?

7. (1 pt) In which area of memory are objects created? ________________

8. (1 pt) When should we define a constant as a static constant? In other words, when should we use static final vs. final while defining a constant?

9. (6 pts) Based on the following class, indicated whether the statements below are valid or invalid. Circle your answer.

   public class Computer { 
   private String make;
   
   public Computer(String makeIn) { make = makeIn; } 
   public void setMake(String makeIn) { make = makeIn; } 
   public static void info() { System.out.println("Computer Sys"); } 
   }

   a. Computer c1 = new Computer("sun");          // VALID / INVALID
      c1.setMake("mars");

   b. Computer c2 = null;                         // VALID / INVALID
      c2.setMake("moon");

   c. Computer c3 = new Computer("earth");       // VALID / INVALID
      Computer.setMake("saturn");

   d. Computer c4 = new Computer("venus");       // VALID / INVALID
      c4.info();

   e. Computer c5 = new Computer("jupiter");     // VALID / INVALID
      c5.make = "uranus";

   f. Computer.info();                            // VALID / INVALID
Problem 2 (8 pts)

Draw a memory diagram showing both the stack and the heap at the moment this program reaches the point marked /* HERE */.

```java
public class MemoryMap {
    public static void filter(StringBuffer state, int by) {
        state.append("now");
        by = 333;
        process(state, by);
    }

    public static void process(StringBuffer first, int last) {
        first = null;
        last = 200;
        /* HERE */
    }

    public static void main(String[] args) {
        StringBuffer orig = new StringBuffer();
        orig.append("cold");
        int val = 100;
        filter(orig, val);
    }
}
```

Stack

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Heap

Stack Bottom

- args
Problem 3 (20 pts)

The method `getMemorySize` takes a string as a parameter that represents the amount of memory present in a high definition camera. The string always starts with the letters H and D, and is followed by a number. The following are some examples of valid strings: “HD100”, “HD2000”, “HD2”, “HD20500”, etc. The method returns the number (an integer) that follows “HD” in the string. For this problem you can assume the caller will provide a valid string or null. If null is provided, the method will throw an IllegalArgumentException with the message “Invalid String”. For this problem:

1. Implement the `getMemorySize` method.
2. Complete the main method provided below so the exception is handled and the message “Invalid String” is printed (using System.out.println) when the exception takes place.
3. Remember that the method `charAt()` returns the character at a particular position in a string.

```java
public class Utilities {
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        // You must complete so exception is handled
        String val = sc.nextLine();
        System.out.println(getMemorySize(val));
    }

    public static int getMemorySize(String memType) {
        // You must write
    }
}
```
Problem 4 (56 pts)

Implement a class named **BillBoard** according to the specifications below.

1. The class has the following private instance variables:
   a. message \(\rightarrow\) String variable
   b. cost \(\rightarrow\) integer variable
2. All the methods in the class are public, except the method **validMessage** that is private. A description of each method follows:
   a. **validMessage** \(\rightarrow\) Takes as parameter a string. The method returns true if the string parameter represents a valid message. A valid message is different from null and it has at least 3 characters.
   b. **Constructor** \(\rightarrow\) Takes two parameters: a string and an integer. If the string parameter represents a validMessage, the instance variables will be initialized using the parameter values. You must use the previous **validMessage** method to validate the string. If the string parameter is invalid, the message instance variable will be initialized to “NOMessage” and the cost to 10.
   c. **Constructor** \(\rightarrow\) Takes a string as a parameter. It initializes the cost to 20. You **MUST** use “this” in order to call the constructor you previously defined otherwise you will not get any credit.
   d. **Copy Constructor**
   e. **getMessage** \(\rightarrow\) Get method for the message instance variable.
   f. **toString()** \(\rightarrow\) Returns a string with the message followed by the cost (separated by one space).
   g. **equals()** \(\rightarrow\) Two objects are equal if they have the same message. The method should return false if null is provided as a parameter value.